一种逻辑块温度和物理块年龄的磨损均衡算法①

刘 柳，黄德才

(浙江工业大学 计算机学院，杭州 310023)

摘 要：UBIFS 文件系统中由 UBI 子系统实现磨损均衡管理，但研究发现 UBI 现有的磨损均衡策略存在着很大的局限性。提出了一种基于逻辑块温度和物理块年龄的新的磨损均衡算法，称为 LTPA(leb temperature peb age)算法。该算法实现高温逻辑块与青年物理块，低温逻辑块与老年物理块之间的映射。通过局部操作时间的方法预测逻辑块的温度，使用循环队列的方式管理空闲物理块。通过实验比较证明了 LTPA 算法在磨损均衡应用中的优越性。

关键词：UBIFS；UBI；磨损均衡；LTPA；循环队列

A Wear Leveling Algorithm Based on Temperature of the Logic Block and Age of the Physical Block

LIU Liu, HUANG De-Cai

(College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract: UBI subsystem is achieving the management of the wear leveling in the UBIFS file system, but the study found that the wear of the existing equilibrium strategy UBI there are significant limitations. This paper propose a new algorithm which is based on the temperature of physical block and the age of the logic block, that is called LTPA (leb temperature peb age) algorithm. The algorithm maps the hot logical block to the young physical block, and the temperature of the cold logical block is mapped to an old physical block. It takes the operating time method to predict the logic block temperature, and using the circular queue to manage the free physical blocks. The simulation algorithm proves the superiority of the LTPA algorithm in wear leveling application.

Key words: UBIFS; UBI; wear leveling; LTPA; circular queue

1 研究背景

近年来，随着嵌入式技术的发展以及在各种电子产品中的广泛应用，嵌入式系统中数据存储和管理的要求越来越高，闪存文件系统的研究也越来越成为人们重点关注的问题 1。随着闪存容量的加大，闪存文件系统的要求也越来越高，它对闪存的存储管理直接影响着闪存的性能和工作效率。

目前可用于NAND型闪存的文件系统包括集中索引的文件系统和专门为 NAND 闪存设计的文件系统 2。集中索引的文件系统是把闪存模拟成类似硬盘一样的块设备，给上层文件系统提供按块读写的接口 3。它存在着一个很大的缺陷，由于闪存不能进行原地的更新，集中索引文件系统下闪存的更新会触发多次额外的数据搬移操作，造成大量空间的浪费，系统效率下降。目前，一些针对 NAND 闪存设计的基于日志结构的文件系统已经得到了广泛的应用，如 JFFS2，YAFFS 等。它们的设计思想是把整个存储空间分成若干个日志块，通过映射表把这些日志块组成文件系统，这样能减少冗余的写操作，提高空间利用率。

UBIFS 文件系统一个是新兴发展的闪存文件系统，它是 JFFS2 文件系统的后继版本 4，UBIFS 文件系统工作在 UBI 层之上，而 UBI 层又关联着 MTD 层，它是连接 MTD 与上层 UBIFS 文件系统的枢纽 5。UBI 子系统负责解决了闪存中最重要的磨损均衡问题，另

① 收稿时间:2011-04-22;收到修改稿时间:2011-05-16

© 中国科学院软件研究所 http://www.c-s-a.org.cn

外它还能解决坏块管理与位反转等一些闪存中的特有问题。而上层的 UBIFS 文件系统是建立在 UBI 的基础上的一个很好的应用，它只对逻辑擦除块进行操作，负责管理文件索引、日志以及文件系统的写缓存机制，并不用关心物理擦除块的磨损均衡操作，也不用关心坏块处理，具有很好的扩展性。本文对 UBIFS 文件系统中的 UBI 子系统管理的磨损均衡机制进行研究与分析，针对 UBI 现有的磨损均衡方法存在的问题，提出了一种新的基于逻辑擦除块温度和物理擦除块年龄的磨损均衡算法，即 LTPA 算法。通过理论分析和实验证明了该算法在磨损均衡应用中的优越性。

2 UBI磨损均衡算法存在着局限性

研究表明，UBI 现有的磨损均衡算法是很有局限性的。首先 UBI 中通过红黑树来管理物理擦除块，对 UBI 的磨损均衡单元来说，物理擦除块有两种类型：used 和 free。通过一个 ubi_wl_get_peb()函数来管理磨损均衡操作的，该函数实现为某个逻辑擦除块分配相应的物理擦除块，它根据写入数据的类型从 free 红黑树中选择不同擦除次数的物理擦除块。而写入数据类型一共可分为三类，分别是 UBI_LONGTERM、UBI_SHORTTERM 和 UBI_UNKNOWN。UBI_LONGTERM 是长期不被更新的数据，一般为只读数据。UBI_SHORTTERM 是频繁更新的数据，这里主要是日志数据。但是，仅仅通过这三种类型来判断写入数据是“冷”还是“热”是很有局限性的，首先如果标记为 UBI_LONGTERM 的数据在写入前的某个时刻受到某个操作的影响在很短时间内不断的更新数百十次，那么 UBI 原有的磨损均衡方法就不能很好的解决这个由冷数据忽然变成热数据的问题。

其次，UBIFS 中的文件索引的根节点存放在逻辑擦除块 LEB1 和 LEB2 中，而索引节点的数据实际是存放在物理擦除块 PEB1 和 PEB2 中，UBI 通过管理映射表把 LEB1、LEB2 与 PEB1、PEB2 建立相应的映射关系。而存放文件索引的信息的块在操作过程中数据更新是非常频繁的，UBI 的磨损均衡方法不能适应由擦除块中数据频繁更新造成的部分擦除块磨损次数过多的问题。而 superblock 区域中的超级节点存放文件系统参数，这部分区域的内容很少改变，而主区中存放了部分数据节点的区域更新操作也很少。包含这两部分的数据可以看成静态数据，也就是“冷”数据。UBI 的磨损均衡中并没有对长期不被更新的“冷”数据进行很好的管理。

针对以上 UBI 在磨损均衡中存在的问题，本文第 3 节中将提出了一种基于逻辑擦除块温度和物理擦除块年龄的磨损均衡算法，称为 LTPA(leb temperature peb age)，它能很好的适用于 UBI 中，帮助 UBI 更好的解决磨损均衡问题。

3 基于逻辑块温度物理块年龄的LTPA算法

由于 UBI 现有的磨损均衡算法存在着两大局限：一是在实际操作中不能很好的适应由擦除块中数据频繁更新造成的部分擦除块磨损次数过多的问题。二是不能有效实现擦除块中冷热数据的搬移。针对以上 UBI 在磨损均衡操作中遇到的问题。下面提出了一种基于逻辑擦除块温度和物理擦除块年龄的新的磨损均衡算法，即 LTPA(leb temperature peb age)算法，它能很好的适用于 UBI 中，有效的实现磨损均衡的管理。

3.1 LTPA 算法基本思想描述

在 UBIFS 中，上层应用程序抽象的把数据写入到逻辑擦除块中，然后 UBI 子系统通过擦除关联表 EBA 实现逻辑块到物理擦除块的映射。LTPA 算法根据在逻辑擦除块中数据的写频率的不同把逻辑擦除块分成高温和低温两类，数据写频率高的逻辑擦除块称为高温逻辑擦除块，写频率低的逻辑擦除块称为低温逻辑擦除块。同样，物理擦除块按其擦除次数的不同也分为两类，老年和青年。即擦除次数高的物理擦除块被称为老年物理擦除块，擦除次数低的物理擦除块被称为青年物理擦除块。

在磨损均衡条件触发时，通过空闲块的分配策略把高温的逻辑块映射到青年的物理擦除块中，把低温的逻辑块映射到老年的物理擦除块上。然后把新的映射关系加入到 EBA 中。如图 1 是 LTPA 算法的基本流程图，当文件数据更新，即上层的应用程序向某逻辑擦除块写入数据时，通过局部操作时间的方法对该逻辑擦除块的数据写频率进行预测，如果此逻辑擦除块数据的写频率较高，那么该逻辑擦除块被定义为高温逻辑擦出块，若此逻辑擦除块数据的写频率较低，则该逻辑擦除块被定义为低温逻辑擦出块。然后通过空闲块的分配策略实现将高温逻辑擦除块映射到青年的物理擦除块中，将低温逻辑擦除块映射到老年的物理擦除块中。空闲块的分配策略由一个循环队列来管理，

© 中国科学院软件研究所 http://www.c-s-a.org.cn

在循环队列中使用二分查找的搜索策略。循环队列中的每一个元组代表一个抽象的空闲物理擦除块，按擦除次数由小到大排列。即擦除次数最小的物理擦除块作为队首，把擦除次数最大的物理擦除块作为队尾。循环队列使用双向链表的方式来实现。

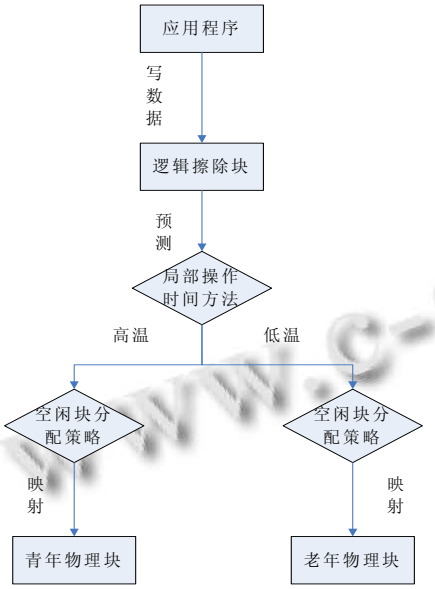

图 1 LTPA 算法的基本流程图

3.2 基于局部操作时间的逻辑块温度预测方法

LTPA 算法的实质是把逻辑擦除块的温度与物理擦除块的年龄联系起来，实现了温度到年龄的映射。那么该算法首先必须判别出逻辑擦除块温度的高低，即逻辑擦除块数据的写频率的大小。本小节通过一种基于局部操作时间的逻辑擦除块的温度预测方法来实现对逻辑块温度的预测。

引入温度 T 的概念来刻画逻辑擦除块中由于数据写入后发生更新操作的时间特性。

$$T(p_i)=\begin{cases}0.9^{\log_2(t-10)} & t>=10\\ 1 & t<10\end{cases} \quad (1)$$

由公式(1)可知 T(pi)表示逻辑擦除块中 pi 页面上的有效数据的温度，其中 t 表示逻辑擦除块中有效数据更新操作的时间和当前系统时间的间隔。当有效数据被部分更新的时候，温度会立即上升，否则温度会随着时间的推移而冷却，从而数据发生改变的可能性大大降低。

知道了逻辑擦除块中每一页的有效数据的温度，就能算出该逻辑擦除块所有页的平均温度以及该逻辑擦除块有效数据页温度的方差。如公式(2)，(3)所示。

$$T(p_m)=\frac{1}{m}\sum_{j=0}^{m-1}T(p_j) \quad (2)$$

$$D(p_t)=\frac{1}{m-1}\sum_{i=0}^{m-1}\left(T(p_i)-T(p_m)\right)^2 \quad (3)$$

$T(p_m)$表示该逻辑擦除块所有页的平均温度，$D(p_i)$表示该逻辑擦除块所有有效数据页温度的方差。定义一个判别逻辑擦除块中各有效页之间温度差异的阈值 DM。当方差 $D(p_i)<DM$ 时，可以认为该逻辑擦除块中各有效数据页的温度变化较小，从而把该逻辑块的平均温度 T(pm)确定为此逻辑块的温度 Tleb。当方差 $D(p_i)>=DM$ 时，则该逻辑块各有效数据页的温度差异较大，说明该逻辑块中存在冷热数据共存的情况，必须实现有效数据的搬移，逻辑块中温度低的数据搬移到另一块温度较低的逻辑擦除块中，把原逻辑擦除块其余有效数据页温度的平均值定为原逻辑块的温度。

定义一个判断逻辑擦除块温度类型的阈值 TM，比较逻辑擦除块的温度 Tleb 与阈值 TM 的大小，当 Tleb>=TM 时，预测该逻辑擦除块为高温逻辑擦除块，当 Tleb<TM 时，预测该逻辑擦除块为低温擦除块。

以上工作描述了一种基于局部操作时间的逻辑擦除块温度预测方法，此算法根据逻辑块的有效页的更新频率来预测该逻辑擦除块温度的高低。预测出逻辑擦除块的温度为 LTPA 算法的后续实现工作做准备。

3.3 基于循环队列的空闲块分配策略

3.3.1 循环队列的基本结构

LTPA 算法中使用循环队列的方式来管理空闲块的分配策略，并在循环队列中使用二分查找的搜索策略。循环队列中的每个元组都是一个空闲物理擦除块的抽象，整个队列由一个双向链表组成，每个元组的内部数据实体为擦除块的擦除次数 erase count，它的值从物理擦除块的擦除计数头部 EHC 中读出。整个循

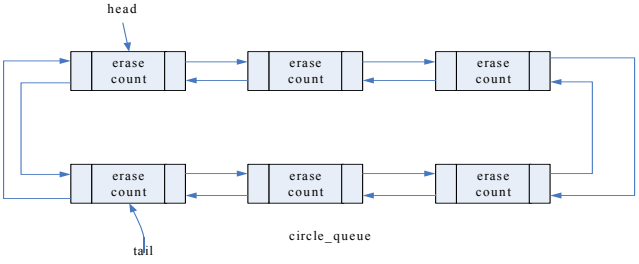

图 2 循环队列结构图

© 中国科学院软件研究所 http://www.c-s-a.org.cn

环队列按 erase count 值由小到大来排序，队首的擦除计数值最小，队尾的擦除计数值最大。队列通过头和尾两个指针 head 和 tail 来操作，实现高效的遍历访问。循环队列的结构如下图 2 所示。

整个循环队列由一个双向链表组成，通过 head 和 tail 两个指针来实现遍历。队列中的每个元组对应一个抽象的物理擦除块，内部唯一的数据实体变量是擦除块的擦除次数 erase count。队列中每个元组抽象的数据结构如下：

```
struct peb_entry_queue {
int erase_count;   /*记录物理擦除块的擦除次数*/
peb_entry_queue * next; /*后继指针，指向下一个结构体对象 peb_entry_queue */
peb_entry_queue * pre; /*前驱指针，指向前一个结构体对象 peb_entry_queue */
} __attribute__((packed));
```

设定循环队列 circle_queue 元组的上限为 MAXSIZE，一般把 MAXSIZE 设定为闪存中总物理擦除块数量的一半，即 MAXSIZE=1/2*total_eraseblock；当循环队列中空闲块的总数 N<=MAXSIZE 时，基于循环队列的空闲块分配策略有效，若 N>MAXSIZE 时，空闲块数量过多，系统负载较小，循环队列使用条件不满足，磨损均衡触发条件不足，系统可采用 UBI 原有的磨损均衡操作对空闲块进行管理。

3.3.2 循环队列实现空闲块分配的工作过程

前面 3.3.1 小节中已经介绍了循环队列是由一个双向循环链表组成，物理块中的元组通过擦除次数由小到大顺序排列。当上层应用程序向某个逻辑块中写入数据时，若通过局部操作时间的方法预测出逻辑块为高温块，即写入数据的更新频率非常频繁，需要取擦除次数较少的青年物理块与高温逻辑块建立映射关系。该算法从空闲块的循环队列中取出队首元组对应的物理块，与之建立映射关系。因为 head-> erase_count 表示擦除次数最小的空闲块。通过该算法能一次遍历取出擦除次数最小的空闲物理块，然后与高温逻辑块建立映射关系，并把映射关系添加到映射表 EBA 中，同时，在队列中重新设置头元素，相应的数据结构操作为 tail->next = head->next，head = head->next，head->next->pre = tail。同样的，如果预测出逻辑块为低温块时，算法从空闲块的循环队列中取出队尾元组对应的物理块，与之建立映射关系。tail->erase_count 表示擦除次数最高的空闲块，算法遍历一次性取出擦除次数最高的空闲块，然后与低温逻辑块建立映射关系，并把映射关系添加到映射表 EBA 中。设置队列的尾元素，相应的数据结构操作为 tail->pre->next = head，head->pre = tail->pre，tail = tail->pre。

LTPA 算法在预测了逻辑块的温度后，用循环队列方式获取擦除次数最高或最低的物理块的匹配次数比 UBI 中默认的用 free 红黑树的方法来获取的匹配次数要少很多。在此种情况下，循环队列的空闲块分配策略比红黑树的空闲块分配策略效率要高。

3.3.3 循环队列空闲块分配的搜索策略

LTPA 算法当磨损均衡条件触发时，系统回收一个脏块，需要把它擦除使之成为一个空闲的物理块。若脏块中包含有效数据，则首先需要进行有效数据的转移，转移的方法可以按照本节前面介绍的方法。当有效数据成功进行转移后，原脏块中的有效数据也变成了无效数据，此时，该脏块中所有的数据都是脏数据，脏块可以开始进行回收操作。回收时首先将脏块擦除，使之成为空闲的物理块，当物理擦除块被擦除后，新的擦除次数被写入该空闲物理块的擦除计数头部中。内存中把该空闲块抽象成一个结构体 struct peb_entry_queue 定义的对象 new_peb，然后把 new_peb 加入空闲块的循环队列中。

把空闲物理块的结构体对象加入循环队列中时，采用二分查找的搜索策略确定该结构体对象加入循环队列的具体位置。循环队列如上图 2 所示，详细操作如下：假定循环对列中共有 N 个元组，即原来有 N 个空闲块。定义一个 struct peb_entry_queue 类型的指针 p，使 p 指向循环队列的头节点 head，然后把指针 p 移动到空闲块循环队列的(N+1)/2 处，比较 p-> erase_count 与 new_peb->erase_count 的大小，若(new_peb->erase_count)>(p-> erase_count)，把指针 p 移动到循环队列后半部分的(N+1)/2 处继续比较，若(new_peb ->erase_count)<(p-> erase_count) ，把指针 p 移动到循环队列前半部分的(N+1)/2 处继续比较。这样不停的采用递归的二分查找方法，直到使该物理块的擦除次数处于循环队列中某两个物理块擦除次数之间时，将 new_peb 插入到 2 个结构体对象之间。

上述的操作循环队列的二分查找算法匹配次数接近 logN，由于指针 p 在寻址过程中时间消耗可以忽略不计，因为 p->next 或 p->pre 操作消耗的代价远远小于比较的消耗。所以基于循环队列的二分查找算法的

© 中国科学院软件研究所 http://www.c-s-a.org.cn

最坏情况复杂度为 O(logN)，而 UBI 中默认的 free 红黑树的查找最坏情况复杂度也为 O(logN)，因此在垃圾回收空闲块过程中循环队列的二分查找方法与 free 红黑树的操作效率基本相当。在 LTPA 算法中，当向擦除块写入数据时，建立逻辑块温度与物理块映射后，应用基于循环队列的空闲块分配策略比 UBI 默认的 free 红黑树的分配方法效率要高。所以在 LTPA 中采用循环队列的方式会使磨损均衡操作变得更加高效。

4 LTPA算法与UBI现有算法实验分析比较

本节通过仿真实验把基于 LTPA 的磨损均衡算法与 UBI 现有的磨损均衡算法进行比较，对比它们在随机更新条件下磨损均衡处理的总体情况。通过各块擦除次数的标准差来判断磨损均衡的效果。并通过多次实验比较内存消耗量。

在仿真实验中，我们建立了一个闪存模拟系统。系统模拟了一个容量为 2MB 的闪存仿真模型，假定每个物理页的大小为 2KB，每个闪存块包含 32 个物理页，每片闪存共有 32 个擦除块。已知 NAND 闪存中每块的实际擦除次数上限为 100 万次，仿真时我们把阈值设置成 1000。根据算法的要求，需要不断的向闪存片中写入数据，然后查看各块擦除次数的大致情况。

实验假定 LTPA 算法与 UBI 磨损均衡算法的阈值均为 1000，即当闪存片中擦除次数最大的块和最小块之间擦除次数的差大于 1000 时，触发磨损均衡操作。如下图 3 和 4 分别为 UBI 磨损均衡算法和 LTPA 算法在磨损均衡上的效果。其中横坐标是块的编号，纵坐标是各块的擦除次数。

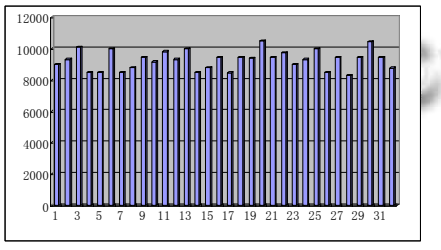

图 3 UBI 磨损均衡算法阈值为 1000 的效果

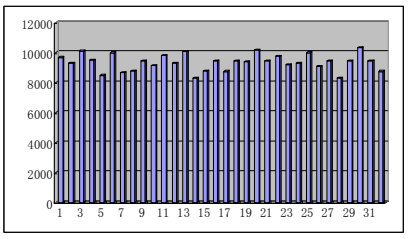

图 4 LTPA 算法阈值为 1000 的效果

当磨损均衡触发条件(即阈值)相同时比较两种算法的磨损均衡效果，比较两个算法物理块擦除次数的标准差与内存消耗如表 1 所示，LTPA 算法与 UBI 的磨损均衡算法在相同的触发条件下查看内存的损耗。

表 1 两种算法擦除次数标准差与内存消耗比较

	UBI 算法	LTPA 算法
擦除次数标准差	605.42	535.03
内存消耗	1.175KB	1.125KB

从表 1 中可以看出 LTPA 算法比 UBI 默认的磨损均衡算法的物理块擦除次数标准差更小，基于循环队列的空闲块分配策略使获取空闲块时比较次数较少，所以内存消耗也较少。

因此，通过实验比较，LTPA 算法比现有的 UBI 磨损均衡算法具有更好的效果，在实际应用中有更大的优越性。

5 总结和展望

UBIFS 是新兴发展的闪存文件系统，它是由上层文件系统和 UBI 子系统两部分组成。UBI 子系统实现闪存的磨损均衡机制的管理。本文针对 UBI 现有的磨损均衡算法存在的局限，提出了一种新的基于逻辑块温度和物理块年龄的磨损均衡算法，称为 LTPA 算法。通过理论分析和模拟实验证明了该算法具有较好的擦除读写性能和磨损均衡效果。另外，闪存的能耗问题闪存研究未来发展的趋势，所以，进一步的工作将考虑在实现磨损均衡的同时尽可能的减小能量消耗，这样的设计思想在实际应用中会变得更有意义。

参考文献

1 Seung HL. An efficient NAND flash file system for flash memory storage. IEEE Transactions on Communications, 2006,55(7):906–912.

2 钟忻,慕春棣.基于闪存的文件系统的实现.计算机工程与应用,2003,14(24):133–135.

3 Chang ML, Lee PC, Chang RC. Managing Flash Memory in Personal Communication Device. ISCE Transactions on Computers,1997,97(7):177–182.

4 韦斯,丁志刚,张伟宏.LINUX 下 UBI 子系统的研究与应用.计算机应用与软件,2010,27(10):68–71.

5 UBIFS Documentation, Nokia & University of Szeged. [2011-3-20].http://www.linux-mtd.infradead.org/doc/ubifs.html

© 中国科学院软件研究所 http://www.c-s-a.org.cn